CLAIMS

What is claimed is:

1. A resource access control mechanism for a multi-thread computing environment, the mechanism being operable:

to manage a sequence of one or more mutexes, wherein the sequence of mutexes is associated with a resource and each mutex may be allocated to one thread; and

when a requesting thread attempts an access to the resource,

to lock a mutex, wherein the locked mutex is allocated to the requesting thread, and

to attempt to lock a previous mutex in the sequence if present, whereby the requesting thread is suspended if the previous mutex is already locked until the previous mutex is unlocked in response to a previous thread finishing access to the resource.

2. The mechanism of claim 1, the mechanism being operable, on attempting to lock a previous mutex in the sequence when the previous mutex is unlocked, to lock the previous mutex on behalf of the requesting thread and then to unlock the previous mutex on behalf of the requesting thread.

3. The mechanism of claim 1, wherein the resource access control mechanism unlocks the mutex allocated to the requesting thread in response to the requesting thread completing access to the resource.

4. The mechanism of claim 1, wherein the mechanism includes an internal mutex operable to protect the locking of the mutex allocated to the requesting thread.

5. The mechanism of claim 1, wherein the resource comprises a print function.

6. The mechanism of claim 1, wherein the sequence of mutexes is held in an array.

7. The mechanism of claim 1, wherein the sequence of mutexes is held in a ring buffer.

8. The mechanism of claim 1, wherein the sequence of mutexes is held in a linked list.

9. The mechanism of claim 1, wherein the sequence of mutexes is held in a circular linked list.

10. A resource access control program for a multi-thread computing environment, the program comprising program code on a carrier medium, which program code is operable to manage a sequence of mutexes, wherein the sequence of mutexes is assocated with the resource and each mutex may be allocated to one thread and is operable to respond to a call from a thread requesting access to a resource by:

locking a mutex for the requesting thread, wherein the mutex is allocated to the requesting thread; and

attempting to lock a previous mutex in the sequence if present, whereby the requesting thread is suspended if the previous mutex is already locked, until the previous mutex is unlocked in response to a previous thread finishing access to the resource.

11. The program of claim 10, wherein the program code is operable, on attempting to lock a previous mutex in the sequence when the previous mutex is unlocked, to

lock the previous mutex on behalf of the requesting thread and then to unlock the previous mutex on behalf of the requesting thread.

12. The program of claim 10, comprising program code operable to respond to a call from the requesting thread completing access to the resource by unlocking the mutex allocated to the requesting thread.

13. The program of claim 10, wherein the program code is operable to control an internal mutex operable to protect the locking of the mutex allocated to the requesting thread.

14. The program of claim 10, wherein the resource comprises a print function.

15. The program of claim 10, wherein the sequence of mutexes is held in an array.

16. The program of claim 10, wherein the sequence of mutexes is held in a ring buffer.

17. The program of claim 10, wherein the sequence of mutexes is held in a linked list.

18. The program of claim 10,wherein the sequence of mutexes is held in a circular linked list.

19. A computer program product comprising program code on a carrier medium, which program code is operable to manage a sequence of one or more mutexes, wherein the sequence of mutexes is associated with a resource and each mutex may be allocated to one thread, and the program code is further operable to respond to a call from a thread requesting access to the resource by:

locking a mutex for the requesting thread, wherein the mutex is allocated to the requesting thread; and

attempting to lock a previous mutex in the sequence if present, whereby if the previous mutex is already locked, the requesting thread is suspended until the previous mutex is unlocked in response to a previous thread finishing access to the resource.

20. The computer program product of claim 19, wherein the carrier medium comprises a storage medium.

21. The computer program product of claim 19, wherein the carrier medium comprises a transmission medium.

22. A computer system comprising:

a processor;

a memory storing a method for controlling access to a resource for a multi-thread computing environment, wherein upon execution of said method on said processor said method comprises:

managing a sequence of one or more mutexes, wherein the sequence of mutexes is associated with the resource and each mutex may be allocated to one thread;

receiving a request from a thread to access the resource;

locking a mutex in the sequence for the requesting thread, wherein the mutex is allocated to the requesting thread; and

attempting to lock a previous mutex in the sequence if present, whereby the requesting thread is suspended if the previous mutex is already locked until the previous mutex is

unlocked in response to a previous thread finishing access to the resource.

23. The computer system of claim 22, further comprising, after attempting to lock a previous mutex in the sequence:

locking the previous mutex on behalf of the requesting thread when the previous mutex is unlocked; and

unlocking the previous mutex on behalf of the requesting thread when the previous mutex is locked on behalf of the requesting thread.

24. The computer system of claim 22, further comprising:

unlocking the mutex allocated to the requesting thread in response to the requesting thread completing access to the resource.

25. The computer system of claim 22, further comprising:

locking an internal mutex operable to protect the locking of the mutex allocated to the requesting thread.

26. The computer system of claim 22, wherein the resource comprises a print function.

27. The computer system of claim 22, wherein the sequence of mutexes is held in an array.

28. The computer system of claim 22, wherein the sequence of mutexes is held in a ring buffer.

29. The computer system of claim 22, wherein the sequence of mutexes is held in a linked list.

30. The computer system of claim 22, wherein the sequence of mutexes is held in a circular linked list.

31. The computer system of claim 22, wherein the method stored in the memory comprises a computer program.

32. A method of resource access control for a multi-thread computing environment, the method comprising:

managing a sequence of one or more mutexes, wherein the sequence of mutexes is associated with a resource, and each mutex may be allocated to one thread;

receiving a request from a thread to access the resource;

locking a mutex in the sequence for the requesting thread, and

attempting to lock a previous mutex in the sequence if present, whereby the requesting thread is suspended if the previous mutex is already locked until the previous mutex is unlocked in response to a previous thread finishing access to the resource.

33. The method of claim 32, further comprising, on attempting to lock a previous mutex in the sequence when the previous mutex is unlocked, locking the previous mutex on behalf of the requesting thread and then unlocking the previous mutex on behalf of the requesting thread.

34. The method of claim 32, further comprising unlocking the mutex allocated to the requesting thread in response to the requesting thread completing access to the resource.

35. The method of claim 32, further comprising managing an internal mutex to protect the locking of the mutex allocated to the requesting thread.

36. The method of claim 32, wherein the resource comprises a print function.

37. The method of claim 32, wherein the sequence of mutexes is held in an array.

38. The method of claim 32, wherein the sequence of mutexes is held in a ring buffer.

39. The method of claim 32, wherein the sequence of mutexes is held in a linked list.

40. The method of claim 32, wherein the sequence of mutexes is held in a circular linked list.